Proof of Concept:

CFFDRS Wildfire Fuel Mapping Prototype

CabinGIS

24/02/2022

Contents

Action

Building on the momentum and ideas of our last meeting, the following pipeline was attempted to produce the wildfire fuel mapping outputs described in the NRC grant "High-Resolution Mapping":

- NRC Grant: https://www.ic.gc.ca/eic/site/101.nsf/eng/00157.html

Using the new cffdrs R-package (Wang et al. 2017; Van Wagner and Pickett 1985; Van Wagner 1987), we developed rasters of forest fuel moisture code and wildfire weather indices (Table 1) that were used to fit the stand-adjusted fine-fuel model (Wotton and Beverly 2007) applied to vegetation rasters classified according to 16 fuel classes of the BC forest fuel typing algorithm @ @ @ perrakis2018british]. REeult rasters were then used to fit the Canadian Forest Fire Behaviour Prediction model from which we drafted raster maps representing Head Fire Index (HFI) and Fire Intensity maps (FI) (Figure 1) for the Okanagan Watershed Basin for the day of June 30th 2021.

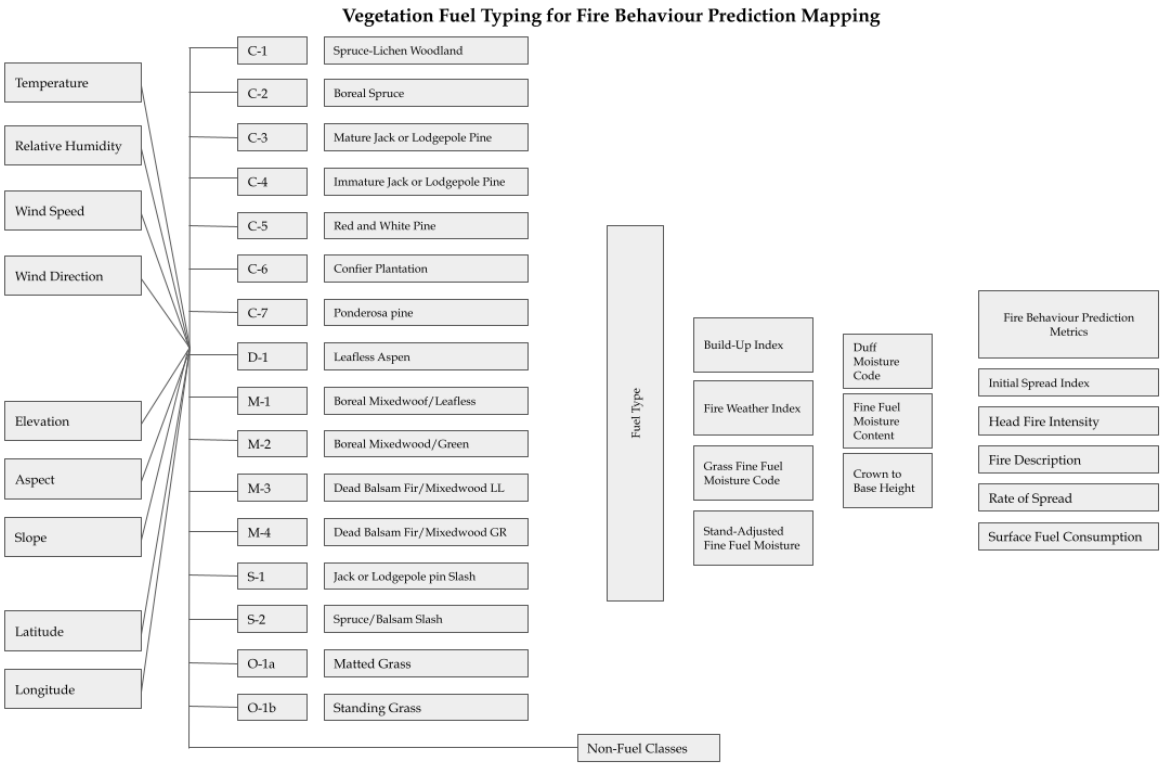


Figure 1: Tentative Workflow

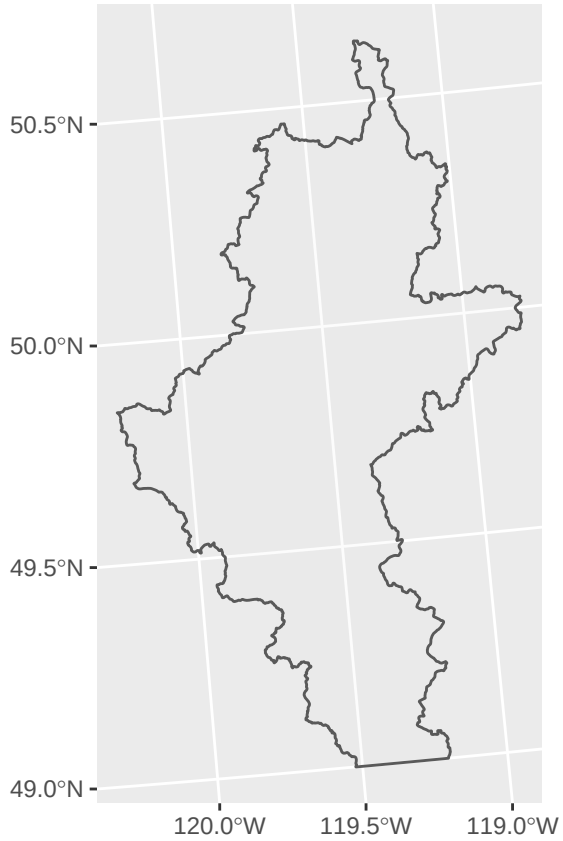
1 Selecting AOI (Okanagan Watershed)

Two input options for selecting AOI were explored below: 1) uploading AOI boundary file and 2) choosing point location as centre of 10km LxW bounding box. For testing boundary file uploads, we downloaded the Okanagan Watershed boundary (FWA ID:212) from the BC Geographic Warehouse which was extracted from the BC Freshwater Atlas Dataset. The aoi feature layer was imported as a simple feature and transformed into sp and spatVector objects for different processing and compatability options. From this initial scoping exercise, the terra route looks like a powerful pipeline (sf > terra > ggmap > elevatr = tmap/gplot), but still needs some figuring out to integrate with the APIs.

```
watershed_okanagan_sf = st_read("./Data/watershed-okanagan-QX.shp")
```

```
## Reading layer `watershed-okanagan-QX' from data source
##   `/Volumes/128GB_WORKD/EFI-FIRE/01_Wildfire-Mapping-CFFDRS-2.0_prototype/Data/watershed-okanagan-QX
##   using driver `ESRI Shapefile'
## Simple feature collection with 1 feature and 6 fields
## Geometry type: POLYGON
## Dimension:     XY
## Bounding box:  xmin: 1413784 ymin: 464311.2 xmax: 1515562 ymax: 647056.5
## Projected CRS: NAD83(CSRS) / BC Albers
```

```
watershed_okanagan_sp = sf:::as_Spatial(watershed_okanagan_sf)
watershed_okanagan_sv = vect(watershed_okanagan_sf)
ggplot(watershed_okanagan_sf) + geom_sf(alpha=0.1) + coord_sf()
```



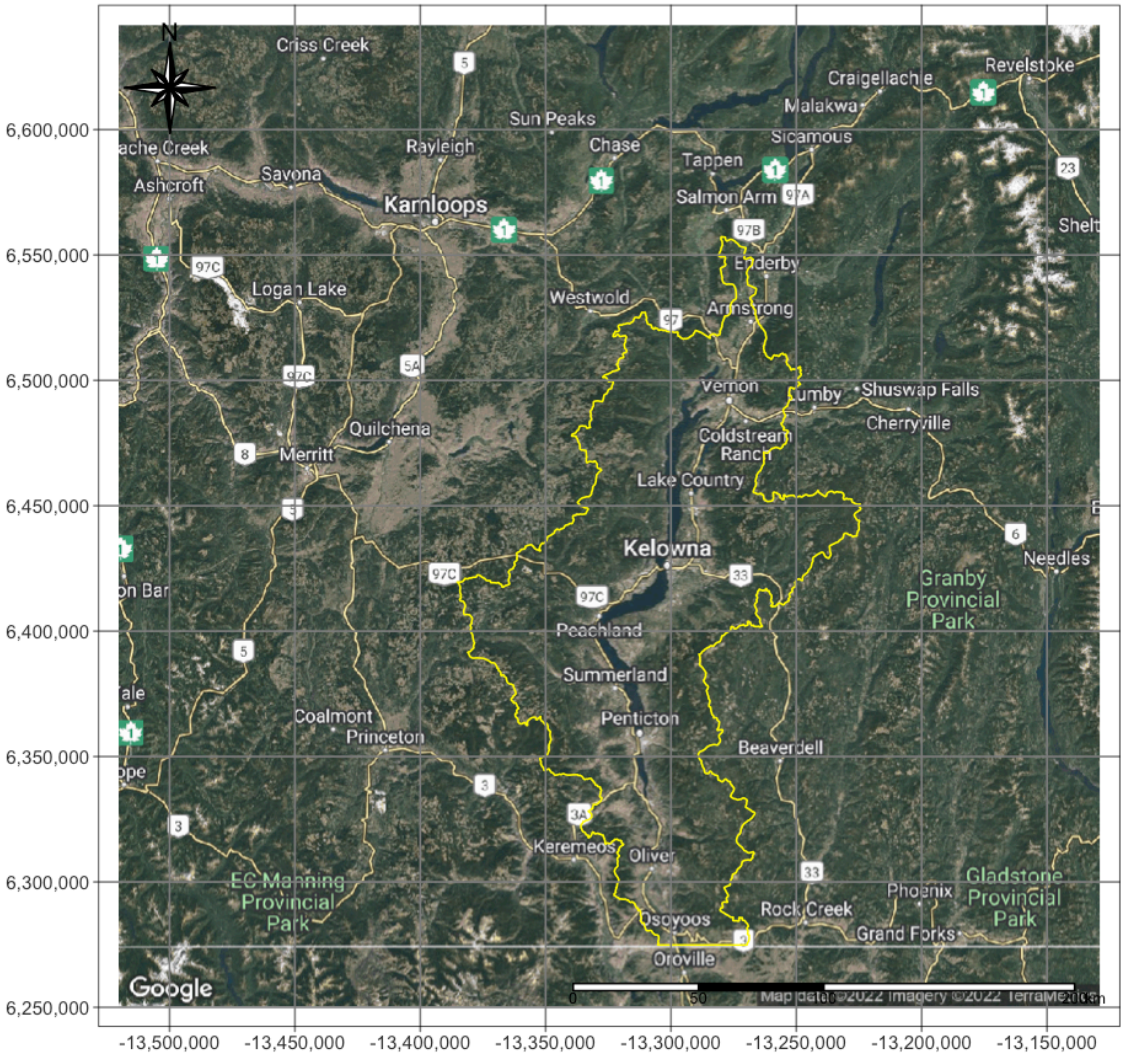
2 Importing Basemaps: Google Cloud Api

For potential GUI-widgets and tools, we tested potential base mapping services from Google Cloud API using the ggmap package and a free API key token set up with personal google account. These required latitude-longitude coordinates as inputs. All data layers were projected to match the Google Pseudo Mercator (EPSG:3857) and then reprojected to CSRS EPSG:3153 for metric analyses. Four basemaps were tested using the API server at a zoom setting of 8 and a latitude and longitude location of -119, 50.0. Still some kinks to work out with the point location input, though seems that this point of interest route is more compatible, especially for working with NASA Power APIs.

```
gmap = get_map(location = c(-119.7, 50.0), maptype = "hybrid", source = "google", crop = FALSE, zoom = 8
gmap1 = ggmap(get_map(location = c(-119.7, 50.0), maptype = "satellite", source = "google", zoom = 8))
gmap2 = ggmap(get_map(location = c(-119.7, 50.0), maptype = "toner-lite", zoom = 8))
gmap3 = ggmap(get_map(location = c(-119.7, 50.0), maptype = "toner-background", zoom = 8))
gmap4 = ggmap(get_map(location = c(-119.7, 50.0), mqptype="terrain-labels", zoom = 8))
```

For manipulating the default 'satellite' basemap layers and overlaying user's aoi boundary, we used Lobo's script (2014) to transform the RGB object from a matrix to a rasterbrick to SpatRaster before applying cartography. The 'Google Pseudo Mercator' (epsg:3857) seems the likely projection used but not much information available on this online. Perhaps more accuracy assessments required in any potential grant objectives. FYI, these chunk outputs take up a good bit of system memory and may cause crashes. For sake of word count, top gmap(1) object was repeated for each gmap output.

```
mgmap <- as.matrix(gmap)
vgmap <- as.vector(mgmap)
vgmaprgb <- col2rgb(vgmap)
gmapr <- matrix(vgmaprgb[1, ], ncol = ncol(mgmap), nrow = nrow(mgmap))
gmapg <- matrix(vgmaprgb[2, ], ncol = ncol(mgmap), nrow = nrow(mgmap))
gmapb <- matrix(vgmaprgb[3, ], ncol = ncol(mgmap), nrow = nrow(mgmap))
rgmaprgbGM <- brick(raster(gmapr), raster(gmapg), raster(gmapb))
rm(gmapr, gmapg, gmapb)
raster::projection(rgmaprgbGM) <- CRS("+init=epsg:3857")
# align crs grids
unlist(attr(gmap, which = "bb"))[c(2, 4, 1, 3)]
rprobextSpDF <- as(extent(unlist(attr(gmap, which = "bb"))[c(2, 4, 1, 3)]), "SpatialPolygons")
raster::projection(rprobextSpDF) <- CRS("+init=epsg:4326")
rprobextGM <- spTransform(rprobextSpDF, CRS("+init=epsg:3857"))
rprobextGM@bbox
extent(rgmaprgbGM) <- c(rprobextGM@bbox[1, ], rprobextGM@bbox[2, ])
plotRGB(rgmaprgbGM)
writeRaster(rgmaprgbGM, file = "./Data/rgmaprgbGM.tif", format = "GTiff", overwrite = TRUE, datatype =
#watershed_okanagan_sp <- spTransform(watershed_okanagan_sp, CRS("+init=epsg:3857"))
gmap_rast = rast(rgmaprgbGM)
RGB(gmap_rast) = c(1,2,3)
crs(gmap_rast) = "EPSG:3857"
crs(watershed_okanagan_sv) = "EPSG:3857"
st_transform(watershed_okanagan_sf, 3857)
# plot objects locally
tm_shape(gmap_rast)+
  tm_rgb(alpha=0.9)+
  tm_shape(watershed_okanagan_sf)+
  tm_borders(col="yellow") +
  tm_compass(type = "8star", position = c("left", "top")) +
  tm_scale_bar(breaks = c(0, 50, 100, 200), text.size = 0.5)+
  tm_grid()
```

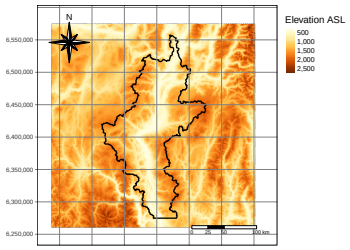


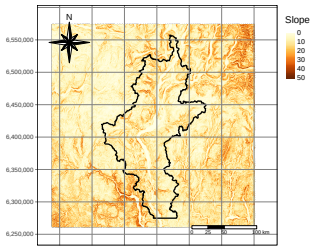
3 Importing Elevation Data

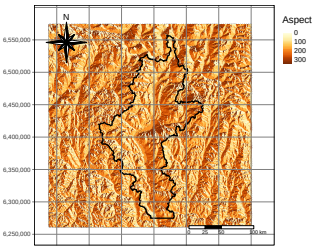
LiDAR data at a resolution of 3 arc seconds was acquired using the 'elevatr' package. As far as we could find, this is one of the few remaining free sources of the SRTM dataset pre-processed at this resolution. It uses their the continuing Mapzen license (@van2001shuttle). DEM data was transformed into a spatRaster and disaggreated from 98m resolution fown to 32m~ resolution. Slope and aspect tasters were calculated using the terra::terrain function with a bilinear interpolation and a rook neighbourhood sampling of 8 adjacent cells. This produced problematic results and the raster processing approach was applied using the deprecated 'slopeAspect' function.

```
ELV = get_elev_raster(watershed_okanagan_sf, z=8)
GS = slopeAspect(ELV, filename = "./Data/GS.tif",
  out='slope', unit='degrees', neighbors=8, overwrite=TRUE)
Aspect = slopeAspect(ELV, filename = "./Data/Aspect.tif",
  out='aspect', unit='degrees', neighbors=8, overwrite=TRUE)
writeRaster(ELV, file = "./Data/ELV", format = "GTiff", overwrite = TRUE, datatype = "INT1U")
raster::projection(ELV) <- CRS("+init=epsg:3857")
```

```
raster::projection(GS) <- CRS("+init=epsg:3857")
raster::projection(Aspect) <- CRS("+init=epsg:3857")
ELV_sr = rast(ELV)
GS_sr = rast(GS)
Aspect_sr = rast(Aspect)
crs(ELV_sr) = "EPSG:3857"
crs(GS_sr) = "EPSG:3857"
crs(Aspect_sr) = "EPSG:3857"
ELV_sr = disagg(ELV_sr, fact=6.8)
GS_sr = resample(GS_sr, ELV_sr, method="bilinear")
Aspect_sr = resample(Aspect_sr, ELV_sr, method="bilinear")
ELV_sr = mask(ELV_sr, watershed_okanagan_sv)
GS_sr = mask(GS_sr, watershed_okanagan_sv)
Aspect_sr = mask(Aspect_sr, watershed_okanagan_sv)
#POI option
tm_shape(ELV)+
  tm_raster(style= "cont", title="Elevation ASL")+
  tm_layout(legend.outside = TRUE)+
  tm_shape(watershed_okanagan_sf)+
  tm_borders(col="black", lwd = 2) +
  tm_compass(type = "8star", position = c("left", "top")) +
  tm_scale_bar(breaks = c(0, 25, 50, 100), text.size = 0.5)+
  tm_grid()
tm_shape(GS)+
  tm_raster(style= "cont", title="Slope")+
  tm_layout(legend.outside = TRUE)+
  tm_shape(watershed_okanagan_sf)+
  tm_borders(col="black", lwd = 2) +
  tm_compass(type = "8star", position = c("left", "top")) +
  tm_scale_bar(breaks = c(0, 25, 50, 100), text.size = 0.5)+
  tm_grid()
tm_shape(Aspect)+
  tm_raster(style= "cont", title="Aspect")+
  tm_layout(legend.outside = TRUE)+
  tm_shape(watershed_okanagan_sf)+
  tm_borders(col="black", lwd = 2) +
  tm_compass(type = "8star", position = c("left", "top")) +
  tm_scale_bar(breaks = c(0, 25, 50, 100), text.size = 0.5)+
  tm_grid()
```








```
# AOI option
tm_shape(ELV_sr)+
  tm_raster(style= "cont", title="Elevation ASL")+
```

```
  tm_layout(legend.outside = TRUE)+
  tm_shape(watershed_okanagan_sf)+
  tm_borders(col="black", lwd = 2) +
  tm_compass(type = "8star", position = c("left", "top")) +
  tm_scale_bar(breaks = c(0, 25, 50, 100), text.size = 0.5)+
  tm_grid()
tm_shape(GS_sr)+
  tm_raster(style= "cont", title="Slope")+
  tm_layout(legend.outside = TRUE)+
  tm_shape(watershed_okanagan_sf)+
  tm_borders(col="black", lwd = 2) +
  tm_compass(type = "8star", position = c("left", "top")) +
  tm_scale_bar(breaks = c(0, 25, 50, 100), text.size = 0.5)+
  tm_grid()
tm_shape(Aspect_sr)+
  tm_raster(style= "cont", title="Aspect")+
  tm_layout(legend.outside = TRUE)+
  tm_shape(watershed_okanagan_sf)+
  tm_borders(col="black", lwd = 2) +
  tm_compass(type = "8star", position = c("left", "top")) +
  tm_scale_bar(breaks = c(0, 25, 50, 100), text.size = 0.5)+
  tm_grid()
```

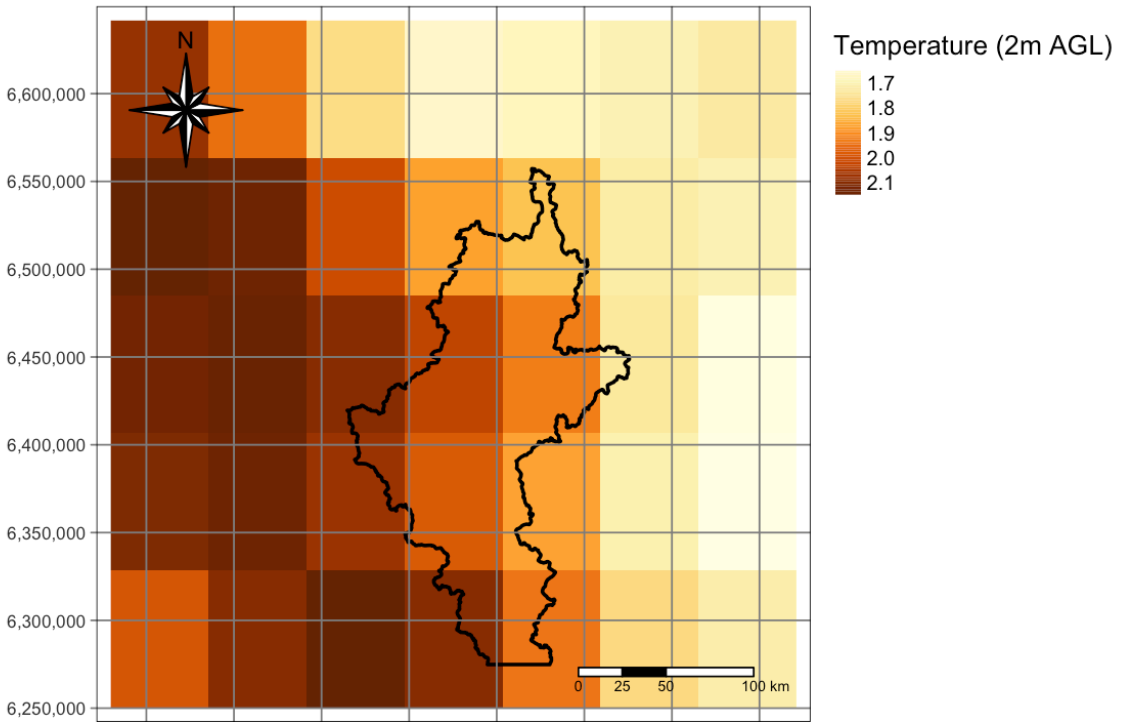
4 Importing Climate Data

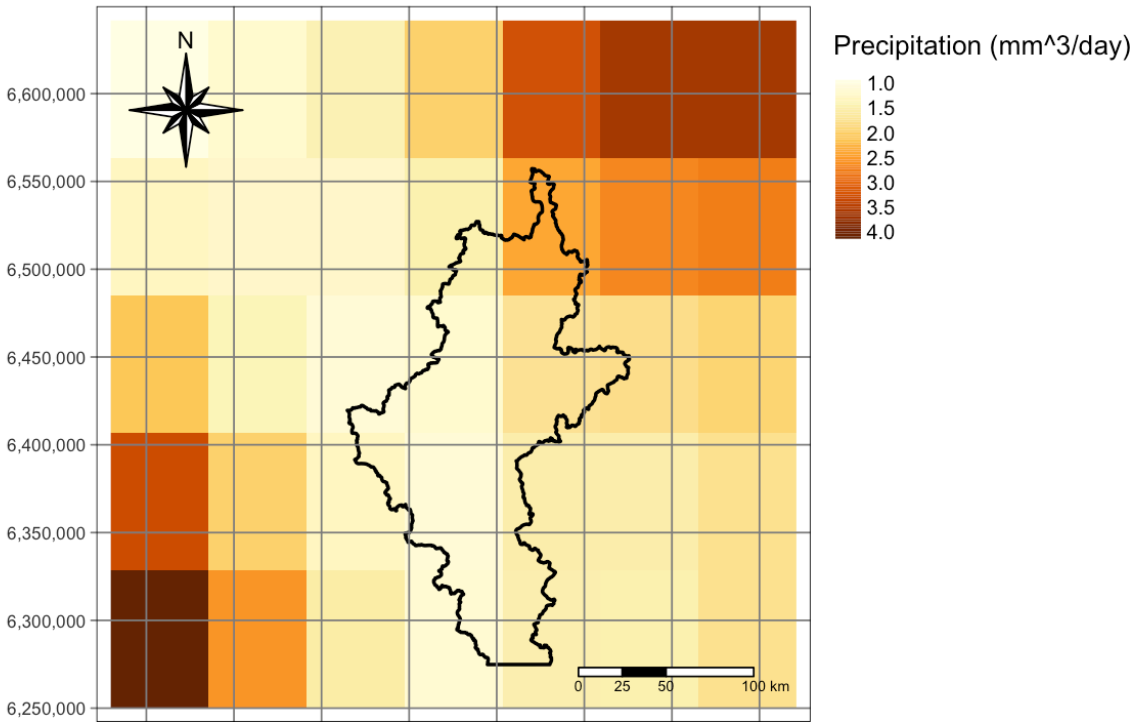
Climate variables were downloaded as NetCDF data from NASA Power platform and read directly into R as spatRraster objects for 1) mean daily temperature at 2m, 2) mean daily precipitation, 3) mean relative humidity, 4) and mean wind speed at 10m. The NASA Power platform supports some very user-friendly API links for static data sources that might be useful for this proposed grant project. REminder tho, some API's prefer dealing with dataframe inputs so might be worht preparing df pipe while still fresh in the head.

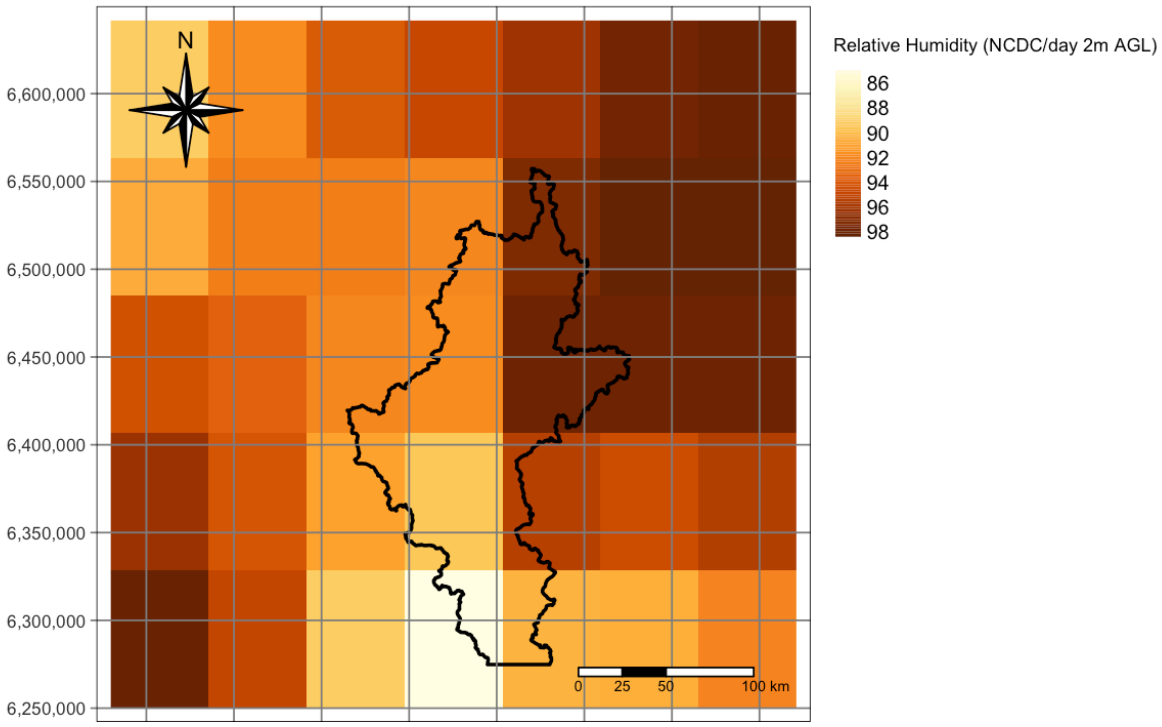
- NASA Power Platform: https://power.larc.nasa.gov/data-access-viewer/

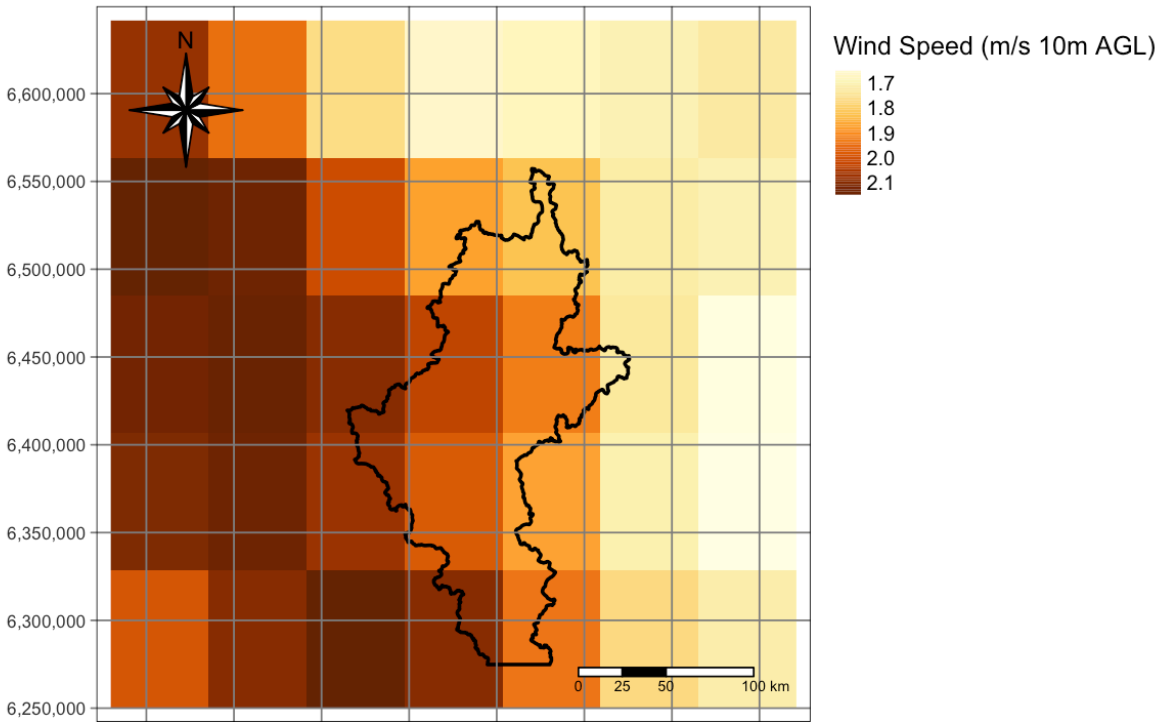
```
temp = terra::rast('NETCDF:"./Data/temp.nc"')
prec = terra::rast('NETCDF:"./Data/precip.nc"')
rh = terra::rast('NETCDF:"./Data/rh.nc"')
ws = terra::rast('NETCDF:"./Data/ws.nc"')
temp_ext = (ext(ELV_sr))
ext(temp) <- temp_ext
temp = mean(temp)
crs(temp) = "EPSG:3857"
prec_ext = (ext(ELV_sr))
ext(prec) <- prec_ext
prec = mean(prec)
crs(prec) = "EPSG:3857"
rh_ext = (ext(ELV_sr))
ext(rh) <- rh_ext
rh = mean(rh)
crs(rh) = "EPSG:3857"
ws_ext = (ext(ELV_sr))
ext(ws) <- ws_ext
```

```
ws = mean(ws)
crs(ws) = "EPSG:3857"

tm_shape(temp)+
  tm_raster(style= "cont", title="Temperature (2m AGL)")+
  tm_layout(legend.outside = TRUE)+
  tm_shape(watershed_okanagan_sf)+
  tm_borders(col="black", lwd = 2) +
  tm_compass(type = "8star", position = c("left", "top")) +
  tm_scale_bar(breaks = c(0, 25, 50, 100), text.size = 0.5)+
  tm_grid()
tm_shape(prec)+
  tm_raster(style= "cont", title="Precipitation (mm^3/day)")+
  tm_layout(legend.outside = TRUE)+
  tm_shape(watershed_okanagan_sf)+
  tm_borders(col="black", lwd = 2) +
  tm_compass(type = "8star", position = c("left", "top")) +
  tm_scale_bar(breaks = c(0, 25, 50, 100), text.size = 0.5)+
  tm_grid()
tm_shape(rh)+
  tm_raster(style= "cont", title="Relative Humidity (NCDC/day 2m AGL)")+
  tm_layout(legend.outside = TRUE)+
  tm_shape(watershed_okanagan_sf)+
  tm_borders(col="black", lwd = 2) +
  tm_compass(type = "8star", position = c("left", "top")) +
  tm_scale_bar(breaks = c(0, 25, 50, 100), text.size = 0.5)+
  tm_grid()
tm_shape(ws)+
  tm_raster(style= "cont", title="Wind Speed (m/s 10m AGL)")+
  tm_layout(legend.outside = TRUE)+
  tm_shape(watershed_okanagan_sf)+
  tm_borders(col="black", lwd = 2) +
  tm_compass(type = "8star", position = c("left", "top")) +
  tm_scale_bar(breaks = c(0, 25, 50, 100), text.size = 0.5)+
  tm_grid()
```


N
6,600,000
6,550,000
6,500,000
6,450,000
6,400,000
6,350,000
6,300,000
6,250,000
0 25 50 100 km
Temperature (2m AGL)
1.7
1.8
1.9
2.0
2.1


N
6,600,000
6,550,000
6,500,000
6,450,000
6,400,000
6,350,000
6,300,000
6,250,000
0 25 50 100 km
Precipitation (mm^3/day)
1.0
1.5
2.0
2.5
3.0
3.5
4.0


N
6,600,000
6,550,000
6,500,000
6,450,000
6,400,000
6,350,000
6,300,000
6,250,000
0 25 50 100 km
Relative Humidity (NCDC/day 2m AGL)
86
88
90
92
94
96
98


N
6,600,000
6,550,000
6,500,000
6,450,000
6,400,000
6,350,000
6,300,000
6,250,000
0 25 50 100 km
Wind Speed (m/s 10m AGL)
1.7
1.8
1.9
2.0
2.1

5 Generating CFFDRS Fuel Type Maps

VRI data was downloaded from imapBC with AOI selected by hand. The data was imported as shapefile.shp and transformed into simple feature for processing. The CFFDRS package has some data formatting requirements. It offers a sample dataset for referring to, including dataframes and raster inputs needed for generating CFFDRS Fire Weather Index maps and CFFDRS Fire Predicted Behaviour maps - see 'test_fwi' and 'test_fpb' presented below.

```
library(cffdrs)
print(as_tibble(test_fwi), n = 10)
```

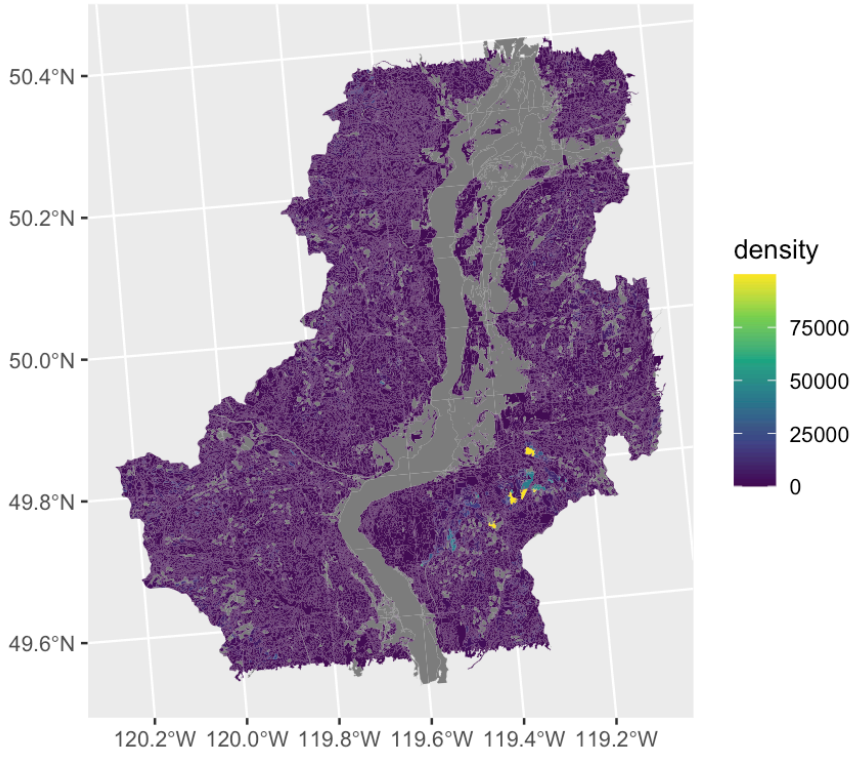
```
## # A tibble: 48 x 9
##     long   lat    yr   mon   day  temp    rh    ws  prec
##    <int> <int> <int> <int> <int> <dbl> <int> <int> <dbl>
##  1  -100    40  1985     4    13  17      42    25   0
##  2  -100    40  1985     4    14  20      21    25   2.4
##  3  -100    40  1985     4    15   8.5    40    17   0
##  4  -100    40  1985     4    16   6.5    25     6   0
##  5  -100    40  1985     4    17  13      34    24   0
##  6  -100    40  1985     4    18   6      40    22   0.4
##  7  -100    40  1985     4    19   5.5    52     6   0
##  8  -100    40  1985     4    20   8.5    46    16   0
##  9  -100    40  1985     4    21   9.5    54    20   0
## 10  -100    40  1985     4    22   7      93    14   9
## # ... with 38 more rows
```

```
print(as_tibble(test_fbp), n = 10)
```

```
## # A tibble: 20 x 24
##       id FuelType   LAT  LONG   ELV  FFMC   BUI    WS    WD    GS    Dj    DO
##    <int> <fct>    <int> <int> <int> <dbl> <int> <dbl> <int> <int> <int> <int>
##  1     1 C-1         55   110    NA  90     130  20       0    15   182    NA
##  2     2 C2          50    90    NA  97     119  20.4     0    75   121    NA
##  3     3 C-3         55   110    NA  95      30  50       0     0   182    NA
##  4     4 C-4         55   105   200  85      82   0      NA    75   182    NA
##  5     5 c5          55   105    NA  88      56   3.4     0    23   152   145
##  6     6 C-6         55   105    NA  94      56  25       0    10   152   132
##  7     7 C-7         50   125    NA  88.8    15  22.1   270    15   152    NA
##  8     8 D-1         45   100    NA  98     100  50     270    35   152    NA
##  9     9 M-1         47    85    NA  90      40  15.5   180    25   182    NA
## 10    10 M-2         63   120   100  97     150  41     180    50   213    NA
## # ... with 10 more rows, and 12 more variables: hr <dbl>, PC <int>, PDF <int>,
## #   GFL <dbl>, cc <int>, theta <int>, Accel <int>, Aspect <int>, BUIEff <int>,
## #   CBH <lgl>, CFL <lgl>, ISI <int>
```

Wotton and Beverly's model of stand-adjusted fine fuel moisture content requires five predictor variables.. Two of these predictors were extracted the the VRI dataset including stand type ('SPEC_CD_1') and stand density (LIVE_STEMS). Applying the MFLNRO VRI-layer fuel-typing algorithm, rough scale criteria were used in a filtering process to classify fuel categories similar to those used in Wotton and Beverley's model. Much more VRI-filtering is possible with the dataset, but not sure if this approach is what NRC are aiming towards especially considering this work was covered by Perrakis et al in 2015.

```
vri2020_sf = st_read("./Data/BCGW_7113060B_1645786298548_3276/VEG_COMP_LYR_R1_POLY/VEG_R1_PLY_polygon.sh
st_crs(vri2020_sf) = crs(watershed_okanagan_sf)
vri2020_sf = st_intersection(st_make_valid(vri2020_sf), watershed_okanagan_sf)

Wotton_fuel_class = vri2020_sf%>%
  mutate(fuel_type = case_when((BCLCS_LV_1 != "V") ~ 0,
  (BCLCS_LV_1 == "V" & BCLCS_LV_4 == "TB") ~ 1,
  (BCLCS_LV_1 == "V" & SPEC_CD_1 == "FD" | SPEC_CD_1 == "FDC" | SPEC_CD_1 == "FDI") ~ 2,
  (BCLCS_LV_1 == "V" & SPEC_PCT_1 <= 80) ~ 3,
  (BCLCS_LV_1 == "V" & SPEC_CD_1 == "PA" | SPEC_CD_1 == "PL" | SPEC_CD_1 == "PLC" | SPEC_CD_1 == "PY") ~
  (BCLCS_LV_1 == "V" & BCLCS_LV_5 =="SP" | SPEC_CD_1 == "SB" | SPEC_CD_1 == "SX" | SPEC_CD_1 == "SW" | S

Wotton_fuel_N = dplyr::filter(vri2020_sf, BCLCS_LV_1 =="N")
Wotton_fuel_decid = dplyr::filter(vri2020_sf, BCLCS_LV_4=="TB")
Wotton_fuel_Df = dplyr::filter(vri2020_sf, BCLCS_LV_1 == "V" & SPEC_CD_1 == "FD" | SPEC_CD_1 == "FDC" |
Wotton_fuel_MW = dplyr::filter(vri2020_sf, BCLCS_LV_1 == "V" & SPEC_PCT_1 <= 80)
Wotton_fuel_PI = dplyr::filter(vri2020_sf, BCLCS_LV_1 == "V" & SPEC_CD_1 == "PA" | SPEC_CD_1 == "PL" | S
Wotton_fuel_PI = dplyr::filter(vri2020_sf, BCLCS_LV_1 == "V" & BCLCS_LV_5 =="SP" | SPEC_CD_1 == "SB" | S
#fuel_attributes = vri2020_sf %>%
# dplyr::select(BCLCS_LV_1, BCLCS_LV_2, BCLCS_LV_3, BCLCS_LV_4, BCLCS_LV_5, SHRB_HT, SHRB_CC, HERB_TYPE
#library(dplyr)
Wotton_fuel_N = 0
Wotton_fuel_decid = 1
Wotton_fuel_Df = 2
Wotton_fuel_MW = 3
Wotton_fuel_PI = 4
Wotton_fuel_SP = 5

density = vri2020_sf["LIVE_STEMS"] %>% mutate(LIVE_STEMS = as.numeric(LIVE_STEMS))
density = rename(density, density = LIVE_STEMS)
ggplot(density) + geom_sf(aes(fill=density), size = 0.0005) + scale_fill_viridis_c()

plot(st_geometry(Wotton_fuel_PI), col="brown", alpha=0.7)
plot(st_geometry(Wotton_fuel_decid), col="green", add=TRUE)
plot(st_geometry(Wotton_fuel_Df), col="brown", add=TRUE)
plot(st_geometry(Wotton_fuel_MW), col="blue", add=TRUE)
plot(st_geometry(Wotton_fuel_PI), col="red", add=TRUE)
plot(st_geometry(Wotton_fuel_SP), col="yellow", add=TRUE)
```




```
# Need to
#### *fwiRaster and samc calculated based on daily climate records*

#### *gfmc and hffmc calculated based on hourly climate records - key to CFFDRSv2.0*

#### *Start date of fire season calculated with fireSeason*

#### *All outputs generated for once-daily calcuylations for the full fireSeason chronologically using
```

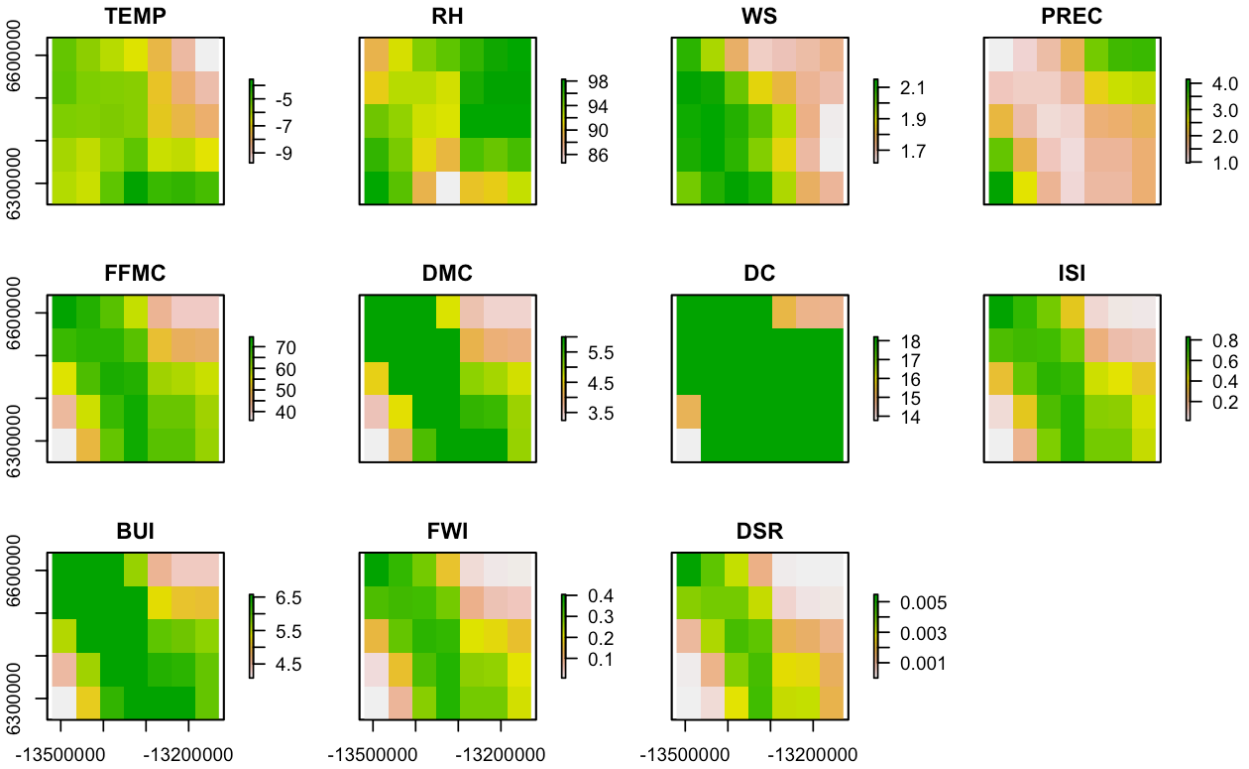
6 Generating CFFDRS Wildfire Weather Maps

Raster stack of interpolated climate predictors was assembled and fitted to the fwiRaster function with 'out="all"' option. This produced raster results for Initial Spread Index (isi), and Build-up Index (bui) and other indices required in FBP calculations. This is as far as we couild get and could only generate these CFFDRS outputs for generic landcover rasters; these scores need to be tuned for each fuel type area.

```
temp = raster::raster(temp)
prec = raster::raster(prec)
rh = raster::raster(rh)
ws = raster::raster(ws)
names(temp) = 'temp'
names(prec) = 'prec'
names(rh) = 'rh'
names(ws) = 'ws'
stack = stack(temp, rh, ws, prec)
names(stack)
fwi_outputs = fwiRaster(stack, out = "all")
```

```
plot(fwi_outputs)
ffmc = raster(fwi_outputs, layer=5)
dmc = raster(fwi_outputs, layer=6)
dc = raster(fwi_outputs, layer=7)
isi = raster(fwi_outputs, layer=8)
bui = raster(fwi_outputs, layer=9)
fwi = raster(fwi_outputs, layer=10)
dsr = raster(fwi_outputs, layer=11)

#define mcF, mcD, ex.mod intermediate functions
mcF<-function(ffmc){
  147.2*(101-ffmc)/(59.5+ffmc)}
mcD<-function(dmc) {
  20+exp(-(dmc-244.72)/43.43)}
ex.mod<-function(s1, s2, s3, ffmc, dmc) {
  exp(s1+s2*log(mcF(ffmc))+s3*mcD(dmc))}

#define stand-adjusted mc function
#FFMC, DMC, stand, density, season
samc<-function(ffmc, dmc, stand, density, season) {
  #Get coefficients
  CoTr1 <-c(
    0.7299,0.0202,0.7977,0.8517,0.7391,
    0.4387,-0.271,0.5065,0.5605,0.4479,
    -0.2449,-0.9546,-0.1771,-0.1231,-0.2357,
    0.1348,-0.5749,0.2026,0.2566,0.144,
    -0.1564,-0.8661,-0.0886,-0.0346,-0.1472,
    -0.84,-1.5497,-0.7722,-0.7182,-0.8308,
    0.1601,-0.55,0.2279,0.2819,0.1693,
    -0.1311,-0.8408,-0.0633,-0.0093,-0.1219,
    -0.8147,-1.5244,-0.7469,-0.6929,-0.8055)
  CoTr2 <- c(
    0.5221,0.6264,0.5042,0.3709,0.4285,
    0.7133,0.8176,0.6954,0.5621,0.6197,
    1.0462,1.1505,1.0283,0.895,0.9526,
    0.8691,0.9734,0.8512,0.7179,0.7755,
    1.0603,1.1646,1.0424,0.9091,0.9667,
    1.3932,1.4975,1.3753,1.242,1.2996,
    0.9495,1.0538,0.9316,0.7983,0.8559,
    1.1407,1.245,1.1228,0.9895,1.0471,
    1.4736,1.5779,1.4557,1.3224,1.38)
  co3 <- 0.002232

  #Create data frame for pulling coefs
  AllCo <-data.frame("co1"=CoTr1, "co2"=CoTr2)

  #Spring and Summer coefs for 'sprummer' model
  co_sp <- AllCo[1:15,]
  co_su <- AllCo[16:30,]


  if(season==1.5) {
    #spring
```

```
    c1.sprD=co_sp[(density*5-4):(density*5), 1]
    c2.sprD=co_sp[(density*5-4):(density*5), 2]
    c1=c1.sprD[stand]
    c2=c2.sprD[stand]
    mc.spr=ex.mod(c1, c2, co3, ffmc, dmc)
    #summer
    c1.sumD=co_su[(density*5-4):(density*5), 1]
    c2.sumD=co_su[(density*5-4):(density*5), 2]
    c3=c1.sumD[stand]
    c4=c2.sumD[stand]
    mc.sum=ex.mod(c3, c4, co3, ffmc, dmc)

    #final 'sprummer' mc calc
    return(mean(c(mc.spr, mc.sum)))

    #for all others - spring, summer or fall
  } else {
    c1a=AllCo$co1[(season*15-14):(season*15)]
    c2a=AllCo$co2[(season*15-14):(season*15)]
    c1b=c1a[(density*5-4):(density*5)]
    c2b=c2a[(density*5-4):(density*5)]
    c1c=c1b[stand]
    c2c=c2b[stand]
    return(ex.mod(c1c, c2c, co3, ffmc, dmc))
  }
}
```



7 Generating CFFDRS Fire Predicted Behavior Maps

References

Van Wagner, CE. 1987. "Development and Structure of the Canadian Forest Fire Weather Index System. Canadian Forestry Service Forestry." Technical Report 35, Ottawa.

Van Wagner, CE, and TL Pickett. 1985. *Equations and FORTRAN Program for the Canadian Forest Fire Weather Index System*. Vol. 33.

Wang, Xianli, B Mike Wotton, Alan S Cantin, Marc-André Parisien, Kerry Anderson, Brett Moore, and Mike D Flannigan. 2017. "Cffdrs: An r Package for the Canadian Forest Fire Danger Rating System." *Ecological Processes* 6 (1): 1–11.

Wotton, B Mike, and Jennifer L Beverly. 2007. "Stand-Specific Litter Moisture Content Calibrations for the Canadian Fine Fuel Moisture Code." *International Journal of Wildland Fire* 16 (4): 463–72.